# **The Stack ADT**

## **Stack Model**

* Stacks are known as LIFO (last in, first out) lists.
* A **stack** is a list with the restriction that insertions and deletions can be performed in only one position, namely, the end of the list, called the **top**.
* The fundamental operations on a stack are
  + push() – inserts a new item to the end of the list
  + pop() – deletes the most recently inserted element (delete the last node in the list)
  + peek() – returns but does not delete the most recently inserted element
* A pop or top on an empty stack is generally considered an error in the stack ADT.
* Running out of space when performing a push is an implementation limit and not an ADT error.

## **Implementation of Stacks**

* Since a stack is a list, any list implementation will do.
* Clearly ArrayList and LinkedList support stack operations; 99% of the time they are the most reasonable choice.
* Because stack operations are constant-time operations, this is unlikely to yield any discernable improvement except under very unique circumstances.
* For these special times, we will give two popular implementations.
* One uses a linked structure and the other uses an array, and both simplify the logic in ArrayList and LinkedList, so we do not provide code.

## **Linked List Implementation of Stacks**

* The first implementation of a stack uses a singly linked list.
  + We perform a push by inserting at the front of the list.
  + We perform a pop by deleting the element at the front of the list.
  + A peek operation merely examines the element at the front of the list, returning its value.

## **Array Implementation of Stacks**

* An alternative implementation avoids links and is probably the more popular solution.
* Mimicking the ArrayList add operation, the implementation is trivial.
* Associated with each stack is
  + theArray: an internal array that will hold the objects for the Stack ADT
  + topOfStack: the index of the current top element, which is −1 for an empty stack (this is how an empty stack is initialized).
* To **push** some element x onto the stack, we increment topOfStack and then set theArray[topOfStack] = x.
* To **pop**, we set the return value to theArray[topOfStack] and then decrement topOfStack.
* Notice that these operations are performed in not only **constant time**, but very fast constant time.
* The fact that most modern machines have stack operations as part of the instruction set enforces the idea that the stack is probably the most fundamental data structure in computer science, after the array.